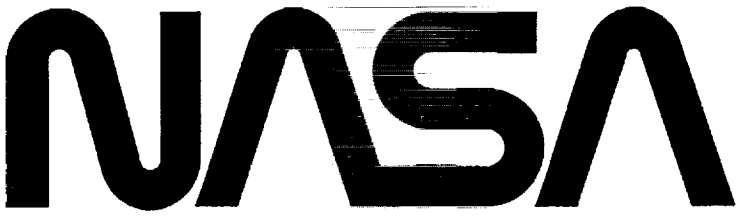
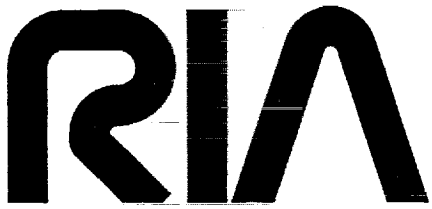


Ames Research Center

LEARNING BY MAKING MODELS

Philip D. Laird
NASA Ames Research Center
Report RIA-88-04-12-0



Artificial Intelligence
Research Branch

RIA-87-11-16-6

Automatic Bayesian Induction of Classes

Peter Cheeseman, James Kelly, Matthew Self, and John Stutz — November 1987

This paper describes a criterion, based on Bayes' theorem, that defines the optimal set of classes (a classification) for a given set of examples. This criterion does not require that the number of classes be specified in advance; this is determined by the data. Tutored learning and probabilistic prediction in particular cases are an important indirect result of optimal class discovery. Extensions to the basic class induction program include the ability to combine category and real-valued data, hierarchical classes, independent classifications and deciding for each class which attributes are relevant.

RIA-88-02-01-01

Knowledge Servers – Applications of Artificial Intelligence to Advanced Space Information Systems

Peter Friedland — February 1988

We have begun a transition from passive information systems which act only to facilitate the storage and retrieval of stereotyped data to far more active and responsive systems which can deal with widely differing forms of human knowledge. Edward Feigenbaum has coined the term 'knowledge servers' to describe this next generation of active information management systems. Among the functions of a knowledge server will be: the ability to store enormous varieties of knowledge; the ability to determine, through natural discourse, the needs of its users; the ability to summarize and pursue complex relationships in its knowledge; the ability to test and critique user hypotheses and suggest previously unseen connections resulting from those hypotheses; and the ability to communicate and collaborate with other autonomous knowledge servers. Because of complexity and variety of information relevant to future major space missions like space station, these missions will act as a driving force and testbed for the knowledge server concept.

RIA-88-04-01-4

AutoClass: A Bayesian Classification System

Peter Cheeseman, James Kelly, Matthew Self, John Stutz, William Taylor, and Don Freeman — April 1988

This paper describes AutoClass II, a program for automatically discovering (inducing) classes from a database, based on a Bayesian statistical technique which automatically determines the most probable number of classes, their probabilistic descriptions, and the probability that each object is a member of each class. AutoClass has been tested on several large, real databases and has discovered previously unsuspected classes. There is no doubt that these classes represent new phenomena.

RIA-88-04-12-0

Learning by Making Models

Philip Laird — April 1988

We propose a theory of learning from unclassified data. The learning problem is that of finding the parameters of a stochastic process that best describes the incoming data stream. Special attention is given to the efficiency of the learning process, similar to Valiant's theory of supervised learning, and in contrast to conventional pattern recognition approaches. Illustrative domains are constructed and analyzed.

Learning by Making Models

Philip D. Laird
NASA Ames Research Center

Moffett Field, CA. 94035

July 14, 1988

Abstract

We propose a theory of learning from unclassified data. The learning problem is that of finding the parameters of a stochastic process that best describes the incoming data stream. Special attention is given to the efficiency of the learning process, similar to Valiant's theory of supervised learning, and in contrast to conventional pattern recognition approaches. Illustrative domains are constructed and analyzed.

1 Introduction

Supervised concept learning occurs when a teacher presents the learner with data labeled according to whether the items are examples or counterexamples of the target concept. The learner must then solve a search problem in a pre-designated concept class. Unsupervised learning omits the teacher, and as a result the nature of the problem changes in an essential way. In this paper we ask what it means for a hypothesis class to be learnable in an unsupervised setting, and if it is learnable, what it means to be efficiently learnable.

For supervised learning, others have given precise mathematical definitions of such terms as "learnable" and "efficiently learnable" (e.g., [32]). This work can be viewed as extending these notions to the unsupervised case. Among the useful results of this work are

- a definite scope for the problem, with clear definitions of such connotative terms as "learn" and "converge" and "approximate".
- the ability to account for noise and temporal dependence among the examples.
- a framework for constructing representations and analyzing their complexity in an unsupervised setting.

First we shall describe informally our vision of unsupervised learning, and relate it to other research. We then formalize the learning problem by defining our model classes and the learning criteria. To illustrate the concepts, we discuss three simple, but non-trivial, model classes. Construction of more powerful models along with algorithms to learn them are topics for subsequent research.

2 Unsupervised Learning: a Model

Let us envision the learner as receiving a stream of input data, either as a continuing sequence of characters (e.g., bits) or as blocks of data, such as vectors or strings. In any case, the data are discrete, both in structure and in time. Initially the learner perceives the input as random, since he/she/it has no prior basis for expectation or prediction. In time, however, patterns are discerned and regularities classified, until the learner is less surprised by what he sees. Ideally such a learner reaches the point where he can predict the next input unit with 100% accuracy. When randomness is inherent in the data, however, this ideal is unattainable without omniscience.

Consider, for example, a stream of input bits generated by independent flips of a biased coin. What form should the learner's predictions take, and how should his predictions be evaluated? The learner might elect to guess the outcome of the next flip, and receive some numerical credit or penalty. But a more general approach – indeed, the one adopted for many statistical pattern recognition problems, and the one we adopt here – is to make predictions in the form of a probability distribution $(\mathbf{P}[H], \mathbf{P}[T])$. Then if an explicit prediction of the outcome of the next coin flip is required, and there are established rewards for good predictions and penalties for

bad ones, we can base our prediction on these criteria using established techniques from decision theory. But these techniques begin with having the probabilities at hand, and the purpose of our learning model is to estimate these probabilities.

Simon [30] defines learning as "changes in the system that ... enable the system to do the same task or tasks ... more efficiently and more effectively the next time." When a system learns probabilities, it may not actually be improving the performance of the system; but it is improving the potential for such performance. So we offer this enhancement to Simon's otherwise useful definition.

In summary, we are treating *unsupervised learning as the inference of a probability distribution for the next input unit.* We shall see that this definition is flexible enough to cover situations where the particular temporal sequence of input units is not of concern, as well as those in which the temporal aspects are significant.

3 Overview of Related Work

Unsupervised learning has been studied experimentally in a variety of situations, including sequence extrapolation, classification and clustering, time series analysis, and adaptive control theory. In each case, the general objective is to infer the underlying structure of the data without benefit of feedback from a teacher. Often it is difficult to compare the various approaches, despite the sense that they are closely related, and we shall not attempt to do so here in any comprehensive way. Instead we identify that work seen as having similar objectives and methods, for the purpose of giving context to the current results.

3.1 Supervised Learning: Convergence

Much of the inductive learning work of the past twenty-five years has centered on the problem of searching a set of descriptions for one which best accounts for the observed data [6, 14, 23, 28]. With guidance from an infallible tutor, the input data serve to eliminate incorrect hypotheses. Consequently, an exhaustive-search technique is effective, but slow, for identifying

a correct description, assuming one exists in the space of descriptions. We shall say that a class of descriptions is *identifiable* (or *learnable in the limit*) if there is a procedure that takes examples from an arbitrary, but exhaustive, presentation and outputs an infinite series of hypothesis descriptions, such that these hypotheses are correct (equivalent to the one responsible for the examples) all but finitely many times.

Absent from most of these studies has been the notion of *convergence* in the Cauchy sense – i.e., progressive improvement in the description as more data are received. This element was provided by a model suggested by Valiant [32], and since extended by others. Good descriptions of the theory (often called learnability theory) are available (e.g., [15]), so we mention here only a few relevant aspects.

Concept: A subset of some universe X, expressible by at least one rule in a fixed class of hypotheses (the *concept class*).

Example: An element of X, positive if it belongs to the target concept, negative otherwise.

Teacher: Selects examples independently and randomly from some probability distribution **P** over the set of examples, and classifies them for the learner as exemplifying or counter-exemplifying the target concept.

Learner: Knows only the concept description class and two parameters, ϵ and δ, stipulating how accurately he is expected to identify the target concept. In particular, the learner does *not* know the distribution **P**.

Learning is viewed as a finite process: the algorithm outputs a concept C and halts. We say that the concept C *has error* ϵ *with respect to* C_*, the target, if the probability (measured by **P**) is ϵ that a random example of C_* is a counterexample for C, or vice versa. Since the selection of examples is a probabilistic process, one cannot require a learning algorithm to achieve a given accuracy with certainty. Instead, one imposes an upper limit δ on the probability that the algorithm chooses a hypothesis with more than ϵ error as a result of seeing an unrepresentative sample.

A concept description class $\mathcal{C}$ is said to be *learnable* if there exists an algorithm $\mathcal{L}$ that, with inputs ϵ, $\delta > 0$, requires only a finite number of classified examples of any target concept $C_* \in \mathcal{C}$ in order to output an approximate concept C. Furthermore, the error of C with respect to C_*

is less than ϵ, with high probability ($> 1 - \delta$). Note that $\mathcal{L}$ must achieve this regardless of the distribution $\mathbf{P}$ of the examples. If no such algorithm exists, the class is not learnable. A useful characterization of what classes are/are not learnable is given in [7].

Often a concept class may be useful on small problems, but when scaled up to express larger concepts, the complexity of the learning problem grows so rapidly that it becomes intractable. To study this, we associate a growth parameter n with the concept class C (e.g., the number of attributes) and characterize the complexity growth of the family C_n as follows.

A concept class C_n is said to be *efficiently learnable* if, for all n,

1. C_n is learnable;
2. the number of examples required by the learning algorithm $\mathcal{L}_n$ is bounded by a polynomial in n, $1/\epsilon$, and $1/\delta$;
3. $\mathcal{L}_n$ processes those examples in time bounded by a polynomial in the size of the examples[1].

Families satisfying the first two properties are said to be *data efficient*. It is not unusual for a family to be known to be data-efficient but not known to be efficiently learnable (e.g., [8, 19]).

Note that this definition limits the number of concepts in C_n to $O(2^{\text{poly}(n)})$, since otherwise we cannot in the worst case even write down the result in polynomial time.

The theory of learnability has attracted much research interest, in large measure because it enables us to analyze and compare the inherent complexity of concept classes and their representations. The same issues are of interest in unsupervised learning; but it is not at all obvious how to formulate the definitions for the unsupervised case.

3.2 Unsupervised Learning: Pattern Identification

Statistical pattern recognition is a mature branch of statistics originating from early work of Pearson and proceeding with particular energy in many directions since the 1960's. The nature of pattern recognition problems

[1] In some studies, the size of the concept is also considered.

in general is to classify data in a meaningful and useful way. Sometimes this means discovering inherent relationships in the data; at other times it means viewing the data the way a human does. Both tutored and untutored problems have been investigated extensively. [10] and [35] each contain readable surveys of various approaches to the field, as well as annotated bibliographies. Here we shall note only the work pertaining to identification of finite mixtures (for which [11] is a good, recent reference) and some of the work on Markov chains, since these are most closely related to the data-modeling ideas presented below.

Finite mixtures. We can imagine a population consisting of a mixture of N component populations over the same vector space X. With the i'th component distribution f_i, we associate a probability p_i. f_i is assumed to be parameterized by some finite set of values λ_i, which we indicate by writing $f_i(\mathbf{x} \mid \lambda_i)$ $(\mathbf{x} \in X)$. The mixture distribution f is then given by

$$f(\mathbf{x}) = \sum_{i=1}^{N} p_i f_i(\mathbf{x} \mid \lambda_i).$$

Intuitively, the process described by such a distribution is one in which a population is selected with probability p_i, and then a vector is drawn from that population according to the probability distribution f_i. The learning problem is to identify N, the p_i, and λ_i from independent random samples of the population distributed according to f. Note that the distributions may overlap – i.e., the same vector may be selected in different ways if it is part of two or more populations.

Three main issues arise in connection with the finite-mixture problem.

1. Is the mixture identifiable? By standard terminology, a mixture is *identifiable* if for no two distinct sets of parameters (N, p's, λ's) does the same mixture distribution result. By this definition, most mixtures of discrete families of distributions (including f's drawn from binomial or uniform distributions) are not identifiable, whereas many mixtures of continuous families (including multivariate normal distributions) are identifiable. Yakowitz and Spragins [34] showed that identifiability is equivalent to linear independence of the family of component distributions.

2. How large a sample of the population is required to identify the mixture components? For statisticians, data efficiency is exhibited when

the metric difference between the true mixture and the sample mixture decreases exponentially with the sample size. A celebrated theorem of Kiefer and Wolfowitz [20] states (roughly): the probability that the difference between the cumulative distribution function (CDF) of the mixture $F(\mathbf{x})$ and the sample CDF $\hat{F}_t(\mathbf{x})$, as measured by the sup norm

$$\sup_{\mathbf{x}} |\hat{F}_t(\mathbf{x}) - F(\mathbf{x})|,$$

exceeds $\epsilon/\sqrt{t}$ is less than $ce^{-b\epsilon^2}$, for all $t > 0$, all $\epsilon > 0$, and some values of c and b that depend on the dimensionality of the vector space. From this it is straightforward to show that the CDF distributions satisfy a learnability property: for any positive ϵ and δ, a sample of size t polynomial in $1/\epsilon$ and $\log 1/\delta$ suffices to ensure that $\hat{F}_t$ is within ϵ of F with probability $1 - \delta$.

3. What algorithm do we use to estimate the parameters from a finite sample? The algorithmic expression of the estimation problem in statistical language is that of finding a consistent estimator for the parameters, given a sample of t data points. This estimator may be expressed as a computable function or as an algorithm. Since most of the results of this field were obtained before complexity theory matured as a mathematical theory, little attention has been paid to the computational complexity of computing the estimator. Furthermore, the sample size required for a given confidence and tolerance is often hard to determine. Yakowitz [33] showed that an effectively computable consistent estimator exists for an identifiable mixture family, subject to some continuity conditions on the parameters.

In a maximum-likelihood estimation approach, a sample of data is obtained and the mixture with the highest probability (likelihood) of producing the sample is selected as the hypothesis. In order to find estimators for the maximum-likelihood mixture distribution, the analytical forms of the distributions $f_i(\mathbf{x} \mid \lambda)$ are substituted into the likelihood function, and an attempt is made to maximize this function over the parameters and the values of the sample data. Except in special cases this produces a closed-form expression that is exponential in the size of the sample. Hence various approximation techniques have been devised in an effort to render the mathematics more tractable.

Bayesian estimation has also received considerable attention as an estimation procedure. Here the idea is to compute the posterior probability $p(\lambda \mid \mathbf{x}_1, \ldots, \mathbf{x}_t)$ from Bayes' rule, the likelihood $p(\mathbf{x}_1, \ldots, \mathbf{x}_t \mid \lambda)$, and

a prior distribution $p(\lambda)$ over the parameters. The choice of priors has been a recurrent source of controversy for Bayesian statistics, but this has been largely resolved with the work of Tribus, Jaynes, Shore, and others. Starting with a minimally informative prior, one can obtain a recursive procedure which uses each successive data point to compute an estimate of a new prior distribution $p(\lambda)$ and and estimate of the parameters λ. Many studies have obtained conditions under which this procedure converges to valid parameters; for example, Aoki [3] obtains such results by applying the Martingale convergence theorem. In general, the Bayesian approach, starting from minimally informative priors, and the maximum-likelihood approach give the same results provided there is sufficient data to distinguish a good choice from a poor one. Bayesian methods are most useful when the sample size is not known to be sufficient for such a determination, and when other knowledge is available to guide the choice of parameters.

Most of this work assumes that the number N of components in the mixture is known. However, Cheeseman and colleagues [9] have applied Bayesian techniques to determine both the mixture distributions and the number of components for normal distribution families, and applied the theory to a number of data sets of different kinds. From the viewpoint of modeling, however, the Bayesian approach has the drawback that a distribution over the parameter space must be computed, and in general this can be costly. A complexity analysis of Bayesian learning would be a useful study.

Markov models. The unifilar Markov-chain model originated with the early work of Shannon, McMillan, Breiman, and others in information theory. The first asymptotic convergence results for the entropy of such a process also date from this work. Markov modeling, subject to a vast array of assumptions and conditions, has been applied to problems ranging from speech recognition to cell biology under the generic label of "hidden Markov models". Such a model consists of a finite Markov chain with, say, n states, and a set of n probability density functions f_i, one for each state in the chain. From its current state, the Markov process makes a transition to some state s, but the actual state of the process is not observable. Instead, a symbol $\mathbf{x} \in X$ is selected according to the distribution f_s and presented to the learner, whose task is to infer the underlying chain and associated

distributions.

Baum and Petrie [4] demonstrate a maximum-likelihood convergence property for chains with a given number of states. Later, in the course of trying to apply this result, Baum and others [5] developed the *re-estimation algorithm* for determining the maximum-likelihood parameters of a given chain. Many applications of this algorithm have been devised, particularly in speech recognition applications (e.g., [22]) where the structure of the underlying chain can be chosen according to linguistic principles.

A different, but closely related, class of Markov models is obtained by associating with each transition of a Markov chain a symbol x from a finite set X. Thus from a given state s, the process selects a transition from s to another state s', according to the probability $p_{s,s'}$ assigned to that transition, and emits the symbol associated with that transition. The learning problem, again, is to infer the structure and parameters of the chain responsible for the observed sequence of symbols. Rudich [25] studied this problem using a maximum-likelihood strategy, with the goal of identifying the minimum-size chain in the limit from an infinite sample, assuming that the transition probabilities are rational. He considers first the case where an upper limit is provided on the number of states of the target chain, and later the case where no such limit is known. Unfortunately his results were only incompletely presented.

In none of the above work, however, is the computational complexity of finding or using the models considered. Indeed, in view of the fact that in actual applications statistically dependent sample data is the norm, not the exception, it is remarkable that learning from other than independent data has been so consistently avoided in the literature.

4 Unsupervised Learnability

This section presents our formal model of unsupervised learning. We need to address four questions:

1. Just what are we learning?

2. How is the target presented?

3. What are the criteria for learning? Specifically, when is a class learnable, and when is it efficiently learnable?

4. Given these criteria, what classes are/are not learnable?

4.1 Model Classes

Recall that unsupervised learning entails extracting predictable aspects from an apparently random input stream. We call the representation of our predictions a *model.*

Before seeing the formal, definitions, we can benefit from an informal preview of the ideas. Let the input stream be $x_1, x_2, \ldots, x_t, \ldots$ taking values in the finite set X. A *model* M is a procedure that, after receiving as input $x_1, \ldots, x_{t-1}$ but before receiving x_t, is capable of computing a (conditional) probability distribution $\mathbf{P}(x_t \mid x_1 \ldots x_{t-1})$ for the next $x_t \in X$. Thus, before seeing x_t, M is able to compute for any $x \in X$ a rational fraction in $[0,1]$, which we interpret as its prediction for the conditional probability $\mathbf{P}(x_t \mid x_1 \ldots x_{t-1})$. To say that M computes a probability distribution means that the values of $\mathbf{P}$ over all x are normalized such that $\sum_{x \in X} \mathbf{P}(x \mid x_1 \ldots x_{t-1}) = 1$. After seeing the actual value of x_t, M updates its "state" to reflect the value x_t, so that it can then compute $\mathbf{P}(x_{t+1} \mid x_1 \ldots x_t)$. This process continues forever.

In addition to these functional requirements, we impose the following efficiency requirements: the time for M to compute any prediction $\mathbf{P}(x_t \mid x_1 \ldots x_{t-1})$ and the time for it to update its state to account for x_t are each bounded as a function of t. Intuitively, this means that M doesn't "slow down" as time goes on; it is able to keep up with the incoming data stream and remain an on-line predictor.

The formal definitions are as follows. Let X be a fixed (and in this paper, finite) set of symbols, and let $Q[0,1]$ be the set of rationals over the interval $[0,1]$.

Definition 4.1 *A* well-parameterized family of distributions *is a finite set* Λ *of parameter values, with an associated algorithm* $\mathbf{P}_{Lambda}$ *that computes a probability distribution on* X *for each* $\lambda \in \Lambda$*:* $\mathbf{P}_{Lambda}(x, \lambda) \in Q[0,1]$,

and $\sum_x \mathbf{P}(x,\lambda) = 1$, *for all* λ.[2]

Example 4.2 Suppose $X = \{0,1\}$. Let Λ be a finite subset of $Q[0,1]$, and for each $\lambda \in \Lambda$, let $\mathbf{P}(x,\lambda) :=$ (if $x = 1$ then λ else $1-\lambda$). This describes a family of Bernoulli distributions (biased coin flips) where the probability is λ of coming up with a "1". By contrast, the family of *all* Bernoulli distributions, for which $\Lambda = Q[0,1]$, is not well parameterized because Λ is infinite. $\triangle$

Definition 4.3 *Fix a well-parameterized distribution family* Λ. *A* model M *is an algorithm that receives as input an infinite stream* $x_1, x_2 \ldots \in X^\infty$ *and in turn outputs a stream* $\lambda_1, \lambda_2 \ldots \in \Lambda^\infty$ *of parameters from* Λ. M *outputs* λ_t *after receiving* x_{t-1} *but before receiving* x_t; *we interpret* $\mathbf{P}(\cdot, \lambda_t)$ *as* M*'s estimate for the conditional probability distribution* $\mathbf{P}[\cdot \mid x_1, \ldots, x_t]$. *Furthermore, the time for* M *to compute and output* λ_t *must be bounded (as a function of* t*).*

A set $\mathcal{M}$ of models over the same X and Λ is referred to as a *model class*. The model class is to unsupervised learning what the concept class is to supervised learning.

Example 4.4 Let Λ be a well-parameterized Bernoulli family, as in Example 4.2. Fix $\lambda \in \Lambda$, and consider the model M that outputs λ for every t. This model corresponds to a view of the world in which the environment is performing a sequence of independent coin flips, yielding 1's and 0's with probability λ and $1-\lambda$, respectively. The class $\mathcal{M}$ of models of this type, one for each parameter in Λ, represents a choice of world-views that agree about the nature of the environment (independent coin-flipping) but disagree about the probabilities. The "real" world may not be any of these, but over a given period of time, one of these models is likely to be the best of its class in accounting for the frequency of 1's and 0's. $\triangle$

When a growth parameter n is given, we have a *model family* $\mathcal{M}_n$, associated with X_n and Λ_n ($n \geq 1$), provided: (i) The algorithm $\mathbf{P}_n$ (corresponding to Λ_n) computes the probability $\mathbf{P}_n(x,\lambda)$ in time bounded by a

[2]When X can be infinite, an additional requirement must be imposed to insure that the algorithm $\mathbf{P}_\Lambda$ runs in time polynomial in the length of x.

polynomial in n; and (ii) each model M in $\mathcal{M}_n$ computes its output λ_t in time uniformly bounded by a polynomial in n. This means that our model families must be able to scale upward at a reasonable rate to handle larger size models and possibly a larger set of input symbols.

Example 4.5 (Bernoulli Models) For $n \geq 1$, let Λ_n be the subset of rationals λ in $Q[0,1]$ whose length $\text{len}(\lambda)$, when written in binary, is at most n bits. Let $\mathcal{M}_n$ be the class of Bernoulli models corresponding to Λ_n, as in Example 4.4. Then $X = \{0,1\}$, Λ_n, and $\mathcal{M}_n$ together define a family of Bernoulli models, with 2^n possible models. The time for a model $M \in \mathcal{M}_n$ to output its prediction is $O(n)$, since all it has to do is output up to n bits. For the same reason, $\mathbf{P}_n$ requires $O(n)$ time to compute probability estimates. $\triangle$

Example 4.6 (Classification Models.) An assumption that is often useful is that successive inputs are statistically independent, i.e.,

$$\mathbf{P}(x_t \mid x_1 \ldots x_{t-1}) = \mathbf{P}(x_t). \tag{1}$$

A model M that incorporates this assumption will predict the same distribution (i.e., output the same parameter λ) for all times t. Since X is fixed in size, any (normalized) assignment of a rational in $Q[0,1]$ to each input x is a model, and any set of such models is a model class. (Example 4.5 is a special case of this idea.)

But suppose the size of X_n is growing exponentially with n — e.g., when n is the number of attributes in an attribute vector. Then models cannot assign probabilities individually to the elements of X, since this requires outputting exponentially many bits. Instead, we need to group vectors into clusters or classes, with the maximum number of such classes growing only polynomially in n. Furthermore, computing the probability for an individual vector in a given class must be fast. For example, an easy way to do this is to assume that all vectors in a class have equal probabilities of occurring next. Finally, we need to limit the size of (i.e., number of bits to represent) the probabilities asociated with the classes, say, to a polynomial in n.

The view of the world painted by these assumptions is that of an environment that randomly selects, independently for each time t, one of a

small (i.e., polynomial-size) number of classes, chooses at random a vector from that class, and presents that vector. (Compare the finite mixture model discussed above.) As with the coin-flip model, the actual environment may not be generated in this fashion, but the best model in the class will probably capture some aspects of the environment, by classifying adjacent vectors into a few (hopefully meaningful) classes.

Later we shall define and analyze a particular family of classification models. △

Example 4.7 (Markov Chain Models.) One of the simplest ways to relax the independence assumption for inputs is to use Markov-dependent random variables. In this case, the total dependence of the t'th prediction on the previous $t-1$ inputs can be reduced to dependence on one of a finite set S of values ("states"). Eq. (1) above can be revised as follows:

$$\mathbf{P}(x_t \mid x_1 \ldots x_{t-1}) = \mathbf{P}(x_t \mid s_{t-1}), \tag{2}$$

where the state $s_{t-1} \in S$ captures all the dependency on $x_1, \ldots, x_{t-1}$.

For simplicity, let $X = \{0,1\}$. A Markov chain[3] over X consists of a (finite) set S of states, one of which is designated the current state; a deterministic transition function δ mapping each state s and input value x into a new state s'; and probabilities $p(1 \mid s)$ and $p(0 \mid s)$ for these transitions summing to 1.

A Markov model M is a program based upon such a Markov chain. Before the t'th input bit arrives, it uses the input x_{t-1} and current state s_{t-1} to compute the next state $s_t = \delta(s_{t-1}, x_{t-1})$, and to output the probability $p(1 \mid s_t)$. The parameterized distribution family Λ is a Bernoulli family (Example 4.4); if M outputs p, this corresponds to a prediction of p for the input 1 and $1-p$ for the input 0. But unlike models that view the environment as a sequence of independent events, M may output different values of p at different times. Since we are interested in modeling environments that are in equilibrium, we restrict our models to those whose Markov chains are strongly connected – i.e., indecomposable, with no transient states.

As a growth parameter n, we typically count the maximum number of states; but we must also limit the length in bits of the transition probabil-

[3] More precisely, a finite function of a homogeneous, discrete time, discrete state, unifilar Markov process.

ities. A bound of length n on probabilities for an n-state machine is quite liberal since it allows exponentially small probabilities. Thus one family $\mathcal{M}_n$ of Markov models corresponds to a fixed set X, the family Λ_n in Example 4.4, and the class of Markov chains with at most n states and with transition probabilities in Λ_n.

It is also possible to allow the size of the input stream X to increase in size with n. In this situation, if X_n grows exponentially with n, then to preserve efficiency we must classify (i.e., group) the values of X as we did in Example 4.6. Each state may correspond to a different classification model, described by a parameter in Λ_n discussed in Example 4.6. (Compare the hidden Markov models discussed above.) In this paper, however, we shall consider X to be fixed.

Δ

To summarize the definitions given in this section, model classes are classes of *stochastic processes*, with special attention given to their computational efficiency. Adopting a model class limits the choice of models we may consider for predicting the future, but in return ensures that we will be able to make predictions online. Classification models arise naturally from the assumption that inputs are independent, whereas Markov models incorporate a simple form of dependence. Naturally, many other types of models are possible as well.

4.2 Presentation

In this work we assume only that the input is a stream $\{x_t,\ t > 0\} \in X^\infty$ of symbols from a finite set X. It is convenient to regard the symbols as arriving one after another at a steady rate. In practice, this rate strongly affects how large an n we can support for our model class $\mathcal{M}_n$ on a given machine.

4.3 Learnability

Having selected a model class $\mathcal{M}$, we then have a learning problem: to find a model in $\mathcal{M}$ that is "optimal" (in some sense) for predicting the input data stream.

How do we decide which model is best? In general there may be no "best" model, because the actual process producing the input may be totally unrelated to the models of our model class. Or imagine, for example, a particularly truculent environment that selects at random a model from $\mathcal{M}$, uses it to generate new data for a random number of steps, and then selects another model. But when in fact the input *really is* the result of a model M in the class, we should certainly expect, disregarding the computational resources required, that some algorithm eventually could identify M, or an equivalent, as the best model. This corresponds to the notion of "learnable in the limit" from supervised concept learning (Section 3.1)[4].

Definition 4.8 *Two models M and M' with outputs in Λ are said to be* equivalent *if, for any input stream, the values of $\mathbf{P}(x, \lambda_t)$ and $\mathbf{P}(x, \lambda'_t)$ are the same for all t and x – i.e., the probability distributions predicted by the two models are always identical, even when their outputs λ_t and λ'_t are different.*

Definition 4.9 *A model class $\mathcal{M}$ is* learnable in the limit *if there exists a procedure $\mathcal{A}$ with the following properties:*

1. *In response to the stream of inputs $\{x_t,\ t \geq 1\} \in X^\infty$, $\mathcal{A}$ outputs a stream of models $\{M_t,\ t \geq 1\}$ in $\mathcal{M}^\infty$. It is often convenient, but not necessary, to consider M_t as the response of $\mathcal{A}$ to the input x_{t-1}.*
2. *Let M be any model in $\mathcal{M}$, and let the input stream be generated according to the probability distribution determined by M. Then with probability one, all but finitely many of the models M_t output by $\mathcal{A}$ are equivalent to M.*

For example, an algorithm modeling the input stream as a Markov process would specify, following each input x, the states, transitions, and transition probabilities for some Markov process in the class $\mathcal{M}$. Assuming the input stream is generated by a process in $\mathcal{M}$, this algorithm should eventually output only Markov processes equivalent to the one responsible for the input. If such an algorithm exists, then $\mathcal{M}$ is learnable in the limit.

[4]The ideas of this and subsequent sections can be made somewhat more precise mathematically by viewing models as defining probability measures on X^∞ and using measure-theoretic formalism to express the results. For simplicity we shall not do so.

It is difficult to imagine a weaker condition than "learnable in the limit" and still view a model class as learnable.

A model that is "way off" in describing the input will over time predict very small probabilities for the inputs that actually occur. By contrast, a better model finds that its predictions come true more often, in that the probabilities for observed inputs are higher. The *likelihood* is a well-known measure of the quality of predictions over the history of the process.

Definition 4.10 *Let M be a model and $X^t \equiv x_1 \ldots x_t$ a sequence of t inputs. Let $\lambda_1, \ldots, \lambda_t$ be the parameter outputs of M over this sequence. The* likelihood *of X^t according to M is given by the product*

$$L(X^t \mid M) = \mathbf{P}(x_1, \lambda_1) \ldots \mathbf{P}(x_t, \lambda_t).$$

The likelihood function is useful in proving learnability in the limit. The idea behind this type of proof is as follows. Let the input be generated by $M \in \mathcal{M}$, and let M' be any model in $\mathcal{M}$. We show that with probability one,

$$L(X^t \mid M) \geq L(X^t \mid M') \tag{3}$$

for all but finitely many t. Furthermore, equality holds iff M and M' are equivalent. Thus a learning algorithm that simply outputs the model with the maximum likelihood for the input so far satisfies the learnability requirement.

The following two results, for example, both from the statistics and information-theory literature, are obtained from this type of proof. (We have provided simple proofs in an appendix to this paper.)

Proposition 4.11 *Let X be a finite set and $\mathcal{M}$ a finite class of models predicting statistically independent inputs over X. Then $\mathcal{M}$ is learnable in the limit.*

As an immediate corollary, classification models (with Bernoulli models as a special case) are learnable in the limit.

Proposition 4.12 (see [4]) *Let X be a finite set and $\mathcal{M}$ a finite class of Markov chain models over alphabet X (see, for example, Example 4.7). Then $\mathcal{M}$ is learnable in the limit.*

An interesting question is for what processes Eq. (3) is also a *necessary* condition for learnability. Also, in recent work [2] Angluin defines the notion of a *uniformly approximately computable* sequence of distributions over the integers. Roughly, the sequence $D_0, D_1, \ldots$ is uniformly approximately computable if some total recursive function exists to compute a rational approximation of $D_i(x)$ to within ϵ, for any i, x, and ϵ. She goes on to show that such a sequence is learnable in the limit. One wonders, for example, whether the likelihood property holds for any such sequence.

Infinite convergence results are not especially useful by themselves, but they frequently point the way toward finite algorithms: run the infinite algorithm for a certain period of time, stop it, and see what you've got. Later this idea will be used to obtain efficient learnability results.

4.4 Learning Minimum-size Models

In many situations finding a model is not enough: we want to find a *small* model equivalent to the source. For example, if we are learning classification models, of the many models equivalent to the correct one, we are usually interested in one with the fewest classes. Minimizing the number of classes in the model makes no difference insofar as the predictions are concerned – any model equivalent to the source yields identical predictions at the parameter level. But when we come to interpret the parametric model in terms of higher-level concepts, then splitting classes into small pieces can make it harder to understand their significance. For example, arbitrarily splitting patients with Syndrome X into several classes makes it harder to identify X as a meaningful diagnostic unit. Similarly, the number of states in a Markov chain model conveys information about the temporal behavior of the source; minimizing the number of states needed to describe this behavior tells us quite a bit about the process generating the input.

Much has been written on the process of determining the number of clusters or classes in a classification (e.g., [9, 24]). Within our model of unsupervised learning, there is a direct resolution to this problem, using some basic probabilistic techniques. In order to illustrate the idea, we define two specific model classes and present algorithms that learn in the limit a smallest model equivalent to the source model.

A simple classification-model family. Let r be a positive integer, and X be the set of ordered n-tuples (vectors) $x_1, \ldots, x_n$, with each element ("attribute") x_i taking integer values in the set $\{0, \ldots, r-1\}$. Thus the cardinality of X is r^n. We are interested in models that partition X into a collection of up to k disjoint subsets $S_1, \ldots, S_k$. We also require that these subsets be orthogonal hyperboxes, defined by a constraint of the form $i \leq x_i \leq j$ for each attribute $(0 \leq i, j \leq r)$. In two dimensions, for example, the $r \times r$ square X is partitioned into rectangles, up to k in number. Associated with each box S_i of the partition is a probability π_i, interpreted as the probability that one of the vectors in set S_i will occur next. If the vector $\mathbf{x}$ is in set S_i, then its probability $\mathbf{P}(\mathbf{x})$ of occurring next is understood to be $\pi_i/|S_i|$. Thus specifying a partition and a normalized set of probabilities for each block has the effect of assigning a probability to each vector in the space.

Initially we shall limit the lengths (in bits) of the set probabilities π_i to k binary bits.[5] Naturally we require that the probabilities of the boxes sum to 1. Furthermore the number of boxes in the partition will be bounded, and for simplicity, we adopt the same constant k for this bound.

A classification model can be encoded in about $kn \log_2 r$ bits[6] by specifying the constraints and probabilities of each of the sets of the partition. Λ_n denotes the set of all such valid specifications. A classification model M_λ outputs the encoding λ of a model before each input value x_t is received. $\mathcal{M}_n$ denotes the family of all such models, as a function of n; we refer to this family as k-RDC (bounded rectangular disjoint classifications). It is easy to verify that Λ_n is well-parameterized by constructing an $O(n)$ algorithm to compute the probabilities of a vector $\mathbf{x}$ from a model parameter λ.

k-RDC is learnable in the limit, by Proposition 4.11. But can we identify a model with the fewest classes? Several techniques have been suggested for this purpose, such as assigning a higher cost to models with greater complexity [24], but we adopt a different approach. The difficulty arises from the random variations in the sampling. Suppose a set $S \subseteq X$ is selected with probability π. Let S_1 and S_2 be disjoint subsets whose union is S, each containing half the points in S; then their selection probabilities

[5] This condition is imposed for the benefit of later complexity results; it is not required for the results of this section.

[6] The default logarithmic base in this paper is e.

are $\pi_1 = \pi_2 = \pi/2$, and the probability of any vector $\mathbf{x}$ in these sets is $p(\mathbf{x}) = \pi/|S|$. But because of statistical fluctuations, this uniformity rarely occurs, and at any time t we are likely to have drawn more vectors from S_1 than from S_2, or vice versa. As a result, there is usually a model in which S_1 and S_2 occur as distinct sets that has a higher likelihood than any model in which S occurs as a unit. Thus the law of large numbers does not prevent maximum-likelihood methods from a bias in favor of more fractionated models.

Our solution to this is to bound the uncertainty with which the parameters of a given model can be estimated. If the vector probabilities $p(\mathbf{x})$ in the subblocks are equal to those of the block they refine to within this uncertainty, we are justified in using the large block instead of the several components, even though doing so may lower the likelihood of the model.

Specifically, let $\{S_1, \ldots, S_p\}$ be a partition of X. Given a sample X^t of t vectors, let us count the fraction of those vectors belonging to S_1, and set $\hat{\pi}_1$ to this fraction; $\hat{\pi}_2, \ldots, \hat{\pi}_p$ are computed similarly. Now we have a model based on the partition $\{S_i\}$ and the parameters $\{\hat{\pi}_i\}$; and it is not hard to show that of all models for this partition, this one has the maximum likelihood for X^t. Repeating this procedure for all possible k-RDC partitions, we obtain a set of *optimal models* of X^t, one for each (admissible) partition of X. Among these will be the maximum-likelihood model(s) for the sample.

In the following algorithm, $\phi(t)$ is any function defined for $t > 0$ such that $\phi(t) \to 0$ and $\phi(t) > \sqrt{\log \log t / 2t}$.

Algorithm 4.13 For each $t > 0$:

1. For all partitions $\Pi_1, \Pi_2, \ldots$ of X, find the optimal models $M_1, M_2, \ldots$ of X^t as discussed above.
2. Of these, a non-empty subset $M_1', M_2', \ldots$ will have maximum-likelihood for X^t. For each of these M_i', we derive a model, minsize(M_i'), as follows:
   (a) Let REF_i be the set of partitions for which Π_i' is a refinement. (Π_i' is the partition that goes with M_i'.)

(b) A partition $\Pi \in \mathrm{REF}_i$ is *unacceptable* if the following condition holds: for some block R of Π and some block S of Π_i' with $S \subseteq R$,

$$\left|\frac{\hat{\pi}_R}{|R|} - \frac{\hat{\pi}_S}{|S|}\right| \geq \phi(t)\left(\frac{1}{|R|} + \frac{1}{|S|}\right). \tag{4}$$

Here, $\hat{\pi}_R$ is the fraction of vectors in X^t belonging to R, and similarly for $\hat{\pi}_s$. Remove all unacceptable partitions from REF_i.

(c) Of the remaining partitions in REF_i, select one with the fewest blocks. minsize(M_i') is defined to be the optimal model for this partition.

3. Among all the optimal models minsize(M_i'), output one with the fewest blocks.

To see why this procedure works, first observe that, for all but finitely many t, with probability one, each of the maximum-likelihood models M_i' will be correct (equivalent to the source), by Proposition 4.11. Thus all we need show is that the algorithm finds a minimal model equivalent to these for almost all t.

Assume then that M_i' is a valid model based on the partition Π_i'. Let Π be a partition refined by Π_i', and M its optimal model. We first argue that, if M is *not* valid, it will almost always be found unacceptable in step 2(b). M is not valid iff there are two blocks S_1 and S_2 in Π_i' in which there are vectors with different probabilities, and some block R in M that contains both S_1 and S_2 (see Fig. 1). By the Strong Law of Large Numbers, the estimates $\hat{\pi}_R/|R|$ and $\hat{\pi}_{S_i}/|S_i|$ converge almost surely to their true values $\pi_R/|R|$ and $\pi_{S_i}/|S_i|$. Let

$$c = \left|\frac{\pi_{S_1}}{|S_1|} - \frac{\pi_{S_2}}{|S_2|}\right|;$$

by hypothesis, $c > 0$. Then $\hat{\pi}_R/|R|$ differs from either $\hat{\pi}_{S_1}/|S_1|$ or $\hat{\pi}_{S_2}/|S_2|$ (say, S_1) by at least $c/4$ for almost all t. But since $\phi(t) \to 0$, the set S_1 will almost always fulfill the condition in Eq. 4, again with probability one, whereupon M will be deemed unacceptable.

Having argued that the algorithm will not continue to combine inequivalent sets, we must convince ourselves that it *will* eventually combine sets that are equivalent. The argument will then be complete.

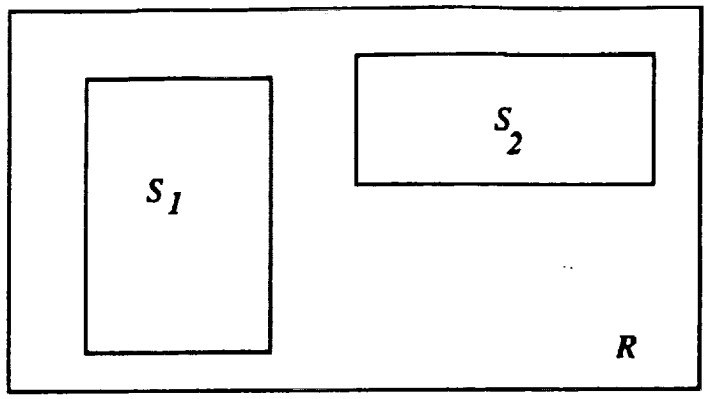


Figure 1: The sets S_1, S_2, and R.

As above, consider a block R in M_i combining blocks S_j $(j = 1, 2, \ldots)$ in M_0 that all have the same vector probabilities. Let π_R be the total probability for the vectors in R and π_j for those in S_j. By the Law of the Iterated Algorithm [12], for a sample of t vectors, the estimate $\hat{\pi}_R$ of π_R is almost always within $\phi(t)$: $|\hat{\pi}_R - \pi_R| < \phi(t)$ a.e. Thus

$$\left|\frac{\hat{\pi}}{|R|} - \frac{\pi}{|R|}\right| < \frac{\phi(t)}{|R|}, \text{ a.e.}$$

Likewise, for the estimates $\hat{\pi}_j$ of S_j we have

$$\left|\frac{\hat{\pi}_j}{|S_j|} - \frac{\pi_j}{|S_j|}\right| < \frac{\phi(t)}{|S_j|}, \text{ a.e.}$$

But since $\pi/|R| = \pi_j/|S_j|$,

$$\left|\frac{\hat{\pi}_j}{|R|} - \frac{\hat{\pi}}{|S_j|}\right| < \frac{\phi(t)}{|S_j|} + \frac{\phi(t)}{|R|}, \text{ a.e.}$$

Hence, by Eq. 4, R is an acceptable agglomeration of the collection S_j. This concludes the proof of

Theorem 4.14 *With probability one, Algorithm 4.13 outputs a k-RDC model of minimum size equivalent to the k-RDC source process for all but finitely many t.*

Minimum-size Markov models. Finding a Markov model of minimum size equivalent to the (Markov) source turns out to be rather easy, owing to the property that every model in our Markov family has a unique smallest equivalent[7] (see [25] for a proof). The family of model classes in which we are interested is that described in Example 4.7, except that X can be any finite set of symbols.

We show that, with no bound on computational resouces, there is an algorithm that identifies the target model exactly, and moreover determines the model with the fewest states equivalent to the target.

The above procedure for finding optimal models for a given partition has a direct analogy for Markov models. Let a *skeleton* be a Markov chain without the transition probabilities specified. Given a string X^t over X of length t, we follow the unique sequence of t transitions, and for each state s and each symbol $a \in X$ count the number of times the a-transition out of that state occurs. Let t_s be the number of times any transition out of s occurs, and $t_s(a)$ the number of a transitions. Then $\hat{p}_s(a) = t_s(a)/t_s$ is a consistent estimator of the transition probability $p_s(a)$, and it is not hard to show that these parameters yield the model with the maximum likelihood for X^t of all models over the same skeleton. Repeating this procedure for all possible skeletons, we obtain a set of *optimal models* of X^t, one for each skeleton of X. Among these will be the maximum-likelihood models. Naturally, we can discard any resulting models which violate the requirements of the Markov class, such as having transient or unreachable states.

The procedure for finding the minimum equivalent model is as follows. For each $t > 0$, with X^t being the input string so far,

1. Determine for each skeleton M the parameters of the optimal model. Discard all such models violating the requirements of the class.

2. Let M' be an optimal model with the maximum likelihood for X^t. Round off all its transition probabilities to n bits, and compute the unique model M'_0 equivalent to M' with a minimum number of states.

3. Output M'_0.

[7] This is not true for Markov models that are not unifilar.

We argue that this simple procedure outputs the minimum-state equivalent to the target model for all but finitely many t (with probability one). By Proposition 4.12, eventually the model M' will (with probability one) be equivalent to the source, since it is a maximum-likelihood model. Since there are no transient states in M', with increasing sample size t, every edge will be traversed an arbitrarily large number of times (with probability one) – i.e., $t_s(a)$ grows without bound. Thus $\hat{p}_s(a)$ converges almost surely to $p_s(a)$. And because the probabilities are bounded below by 2^{-n}, the rounded estimates eventually are exactly equal to $p_s(a)$ (with probability one). Thus the model M' will be correct all but finitely often. And since it has a unique minimization M'_0, that too will be correct all but finitely often. This proves:

Theorem 4.15 *There is an algorithm that, with probability one, outputs a Markov model of minimum size equivalent to the Markov source process for all but finitely many t.*

4.5 Efficient Learnability

Learnability in the limit does not necessarily imply uniform convergence over time – i.e., gradually improving the quality of the model. But, as noted earlier, it is often the case that an in-the-limit algorithm can be truncated and turned into a probabilistic approximation algorithm that runs in finite time. We note that (as with supervised learning) a finite algorithm can usually be expected only to approximate the input source. How do we measure the closeness of such an approximation?

Definition 4.16 *Let $\mathcal{M}$ be a model class. A distance measure d is a non-negative real function on pairs of models $\mathcal{M} \times \mathcal{M}$ such that: (i) for equivalent models, M and $M' \in \mathcal{M}$, $\mathrm{d}(M' \mid M) = 0$; (ii) for inequivalent models, $\mathrm{d}(M' \mid M) > 0$.*

Example 4.17 Let M be the target model, and M' any other model. A strong distance measure is the maximum possible difference in the predicted probabilities of the two models:

$$\mathrm{d}_1(M' \mid M) = \max_{x_t} |\mathbf{P}(x_t, \lambda'_t) - \mathbf{P}(x_t, \lambda_t)|.$$

Another popular distance function is variously known as the cross entropy, informational divergence, or Kullback-Leibler function [18, 21, 29]:

$$\mathrm{d}_2(M' \mid M) = \lim_{t\to\infty} \frac{1}{t} \log \frac{L(X^t \mid M)}{L(X^t \mid M')},$$

assuming the limit exists. For ergodic processes with the likelihood property (Eq. 3), including the the classification and Markov chain models, this is a well-defined distance measure. The fact that this function is not symmetric makes it somewhat more difficult to work with.

Finally, the mean-squared error, generally used only for time-independent processes, is defined by

$$\mathrm{d}_3(M' \mid M) = \lim_{t\to\infty} \sum_{i=1}^{t} \frac{[\mathbf{P}(x_i \mid \lambda') - \mathbf{P}(x_i \mid \lambda)]^2}{t},$$

again assuming the limit exists. For independent processes, this limit is the expected value of the squared difference in predicted probabilities. △

Given such a distance measure d, we divide the efficiency question into two parts. Let $\mathcal{M}_n$ be a (learnable) model family, with n as the growth parameter.

Definition 4.18 *$\mathcal{M}_n$ is said to be* data efficient *(with respect to the distance measure d) if there exists a polynomial f and an algorithm $\mathcal{A}$, taking as parameters n, ϵ, and δ, with the following property: for an input stream generated by any $M \in \mathcal{M}$, $\mathcal{A}$ examines $t \leq f(n, 1/\epsilon, 1/\delta)$ inputs $x_1 \ldots x_t$ and outputs a model $M' \in \mathcal{M}_n$ such that $\mathrm{d}(M' \mid M) \leq \epsilon$, with probability $\geq 1 - \delta$.*

Below, we give examples of model families that are and are not data efficient, with respect to different distance measures.

As with supervised learnability, there are families that we know are data efficient but for which we do not yet have algorithms to find good models from the input data. The ability to find good models quickly from a sufficient sample of the environment is captured by the following definition.

Definition 4.19 *$\mathcal{M}_n$ is said to be* efficiently learnable *if it is data efficient, and the learning algorithm $\mathcal{A}$ runs in time bounded by a polynomial in the total size of the input data.*

Below we analyze the efficiency of learning the three sample model families defined above, the Bernoulli, k-RDC, and Markov families.

4.5.1 Bernoulli Models

The Bernoulli family of models was defined in Example 4.5. There are 2^n possible models in this family; the fact that we can efficiently find an ϵ-approximate model is, of course, a direct consequence of the exponential convergence of sums of independent random variables to their mean.

Assume the environment is a sequence of 1's and 0's generated by a Bernoulli process whose parameter is $\lambda_* \in \Lambda_n$. Let $\mathbf{d}_1$ be our distance measure. Given ϵ and δ, we want to choose $\lambda \in \Lambda_n$ such that $|\lambda - \lambda_*| \leq \epsilon$, with probability $\geq 1 - \delta$.

Define ϵ' to be $2^{-n}/3$ if $\epsilon < 2^{-n}$, or $\lfloor \epsilon \cdot 2^n \rfloor \cdot 2^{-n}$ otherwise. We argue that at most

$$t = \left\lceil \frac{\log_e(2/\delta)}{2\epsilon'^2} \right\rceil \tag{5}$$

inputs are required to obtain an ϵ-approximate parameter λ. (Note that the sample size t is clearly polynomial in n, $1/\epsilon$, and $\log 1/\delta$.) Given this many inputs, let t_1 be the number of 1's in the string X^t. Our algorithm is to determine the proportion t_1/t of 1's and to round this estimate to the nearest multiple of 2^{-n}; this value is output as our hypothesis λ.

To see that t is sufficiently large, consider first the case where $\epsilon \geq 2^{-n}$. Thus $\epsilon' \leq \epsilon$, and by Hoeffding's Inequality [17], in t trials,

$$\begin{aligned} \Pr\left[\left|\frac{t_1}{t} - \lambda_*\right| \geq \epsilon'\right] &\leq 2e^{-2t\epsilon'^2} \\ &\leq \delta. \end{aligned}$$

Thus with sufficiently high probability, the (rational) estimate t_1/t is within ϵ' of λ_*. Consider now the effect of rounding. ϵ' is ϵ reduced to the next lower multiple of 2^{-n}. If $\lambda_* = i2^{-n}$ and $\epsilon' = j2^{-n}$ for some integers i and j, then since with high probability t_1/t is within ϵ' of λ_*,

$$(i-j)2^{-n} \leq t_1/t \leq (i+j)2^{-n}.$$

Rounding t_1/t gives λ, for which we have

$$(i-j)2^{-n} \leq \lambda \leq (i+j)2^{-n},$$

and so $|\lambda - \lambda_*| \leq \epsilon' \leq \epsilon$.

In the other case, when $\epsilon < 2^{-n}$, ϵ' is fixed at $\frac{1}{3}2^{-n}$. Hoeffding's Inequality still guarantees that t_1/t will be within ϵ' with high probability, and rounding will therefore result in $\lambda = \lambda_*$. We have thus proved:

Theorem 4.20 *The family of Bernoulli models is data-efficient, with respect to the* d_1 *distance. Furthermore, when* $\epsilon < 2^{-n}$, *the model is learnable exactly from polynomially many input data, with high probability* ($> 1 - \delta$).

As a corollary, we note that the above algorithm for processing the t data values runs in time polynomial in t. Hence

Corollary 4.21 *The family of Bernoulli models is efficiently learnable (with respect to* d_1).

Suppose next that our model class does not correspond to reality, that the actual probability λ_* is not a multiple of 2^{-n}. Then the algorithm above will still produce a parameter $\lambda \in \Lambda_n$, but as it happens, λ *may not be within* ϵ *of* λ_* because rounding could push it off in the wrong direction. A simple change to the calculation of ϵ' will fix this small problem, but this illustrates that, even if an algorithm correctly finds an ϵ-approximation to any model within the hypothesized model class $\mathcal{M}_n$, that same algorithm *may not produce an* ϵ*-approximation when the data is* not *a result of any model in the class, and may not even find the model in the class that most closely describes the data.*

Suppose we prefer a different distance measure? Obviously, the above results hold for the d_3 distance measure. In the appendix we show that Bernoulli models are still data efficient and efficiently learnable when the d_2 distance measure is used. Our upper bound on the size t of the data sample differs from that given above for d_1 by a small factor.

4.5.2 k-RDC Models

The k-RDC family was defined in Section 4.4. The key feature of this class is that the maximum number k of boxes is bounded, regardless of the number of attributes. While this may seem unreasonable, it is actually a useful strategy: humans make most effective use of a classification if the

number of classes is kept reasonably small. We shall discover that, by bounding the number of classes, our task becomes one of identifying those attributes that are most useful for classification.

Suppose a stream of vectors is being generated by a k-RDC model, and consider how we might go about identifying the source model. Proposition 4.11 and identification-in-the-limit give us a generous hint: eventually a correct model will have a higher likelihood than any incorrect one. The question is, how large a sample is needed before all "bad" models – i.e., ones that are not ϵ-approximations to the source – have significantly lower likelihood, with high probability?

Let M_* be the source k-RDC model, and let M_1 be a k-RDC model based on the partition $\mathcal{R} = \{R_i\}$ and assigning a probability of π_i' to the block R_i. Assume we have a sample X^t of t vectors, randomly generated according to the statistics determined by M_*. Our first result is that we can estimate for each block (box) R_i the total probability for the vectors in that box.

Lemma 4.22 *Fix any $\delta > 0$. Let R be any box in a k-RDC partition. The probability π_R that the next input vector belongs to R can be determined exactly, with probability at least $1 - \delta$, from a sample of t vectors where t is bounded by a polynomial in n and $\log(1/\delta)$.*

PROOF: Our proof will appeal to the previous result for the Bernoulli family. As usual, let M_* be the source model and M_1 any other k-RDC model. First we argue that any box in a k-RDC partition contains a large number of vectors. Imagine how we might partition the entire vector space of r^n vectors into at most k rectangular boxes. In specifying a block of that partition, we select a non-trivial constraint on, say, the first dimension: $x_1 \leq b_1$, where $b_1 < r - 1$. As a consequence of this constraint, the resulting partition must contain at least two blocks. A further constraint, whether of the form $x_1 \geq a_1 > 0$ or a constraint on a different attribute, will mean that the resulting partition must contain at least three blocks, as a consequence of the orthogonal geometry. In fact, each constraint bounding an attribute away from an extreme value (0 or $r - 1$) forces the resulting partition to have another block. Since the number of boxes is bounded by k, there can be at most $k-1$ constraints on any box; hence at least $n-k+1$ attributes of

any box have no constraint whatsoever, and allow the attribute to assume all possible values between 0 and $r-1$. It follows that each box contains at least r^{n-k+1} vectors.

Next, suppose for the moment that R is a block of the target model M_*. How would we estimate the probability π_R? π_R is a multiple of 2^{-k}. Hence estimating π_R is equivalent to learning a Bernoulli model, and since 2^{-k} is constant, π_R can be learned exactly, with high probability, from a small sample, by Theorem 4.20.

Now let R be a block in the partition of an arbitrary model M_1; we argue that the problem of estimating the probability π_R is again a matter of learning a Bernoulli parameter, and that we can do so exactly with high probability. Denote by by X_j the subset $R \cap S_j$ of R contained in the j'th block S_j of M_*. We've already seen that R is big, but X_j must also be big. To see this, note that R is partitioned into at most k rectangles by the various blocks X_j. Each X_j is defined by at most $2k-2$ constraints ($k-1$ for R and $k-1$ for S_j); hence there are at least $n-2k+2$ unconstrained dimensions. Now, either π_R is zero, or for some j, X_j contains vectors occuring with non-zero probability. In the latter case, we can lower-bound π_R as follows.

Suppose $\pi_{S_j} = c_j \cdot 2^{-k}$, where $c_j > 0$. $|S_j|$ can be expressed as

$$\nu_1 \ldots \nu_{k-1} \cdot r^{n-k+1},$$

where the ν_i's are integers in the range $[1, r]$ representing the number of possible distinct values the i'th constrained dimension can assume. Similarly, $|X_j|$ can be written

$$\nu'_1 \ldots \nu'_{k-1} \mu'_1 \ldots \mu'_{k-1} \cdot r^{n-2k-2},$$

where the ν'_i's are the multiplicities for the dimensions that correspond to the ν_i's and the μ'_i's are those for the additional dimensions constrained by R. The probability π_{X_j} of obtaining a vector in X is then given by

$$\begin{aligned} \pi_{X_j} &= \frac{\pi_{S_j}}{|S_j|}|X_j| \\ &= c_j 2^{-k} \cdot \frac{\nu'_1 \ldots \nu'_{k-1} \mu'_1 \ldots \mu'_{k-1}}{\nu_1 \ldots \nu_{k-1} \cdot r^{k-1}} \cdot \frac{r^{n-2k-2}}{r^{n-2k-2}} \\ &= c_j 2^{-k} \cdot A. \end{aligned}$$

(We have introduced the symbol A to stand for the complicated fraction with μ's and ν's). Now, A is a multiple of $1/(r!)^{2k-2}$, and thus π_{X_j} is a multiple of $\beta \equiv 2^{-k}(1/(r!)^{2k-2}) = \text{constant}$. Since $\pi_R = \sum_j \pi_{X_j}$, it too is a multiple of β. Hence learning the parameter π_R is a matter of learning a Bernoulli parameter, which we can do using a sample of size $O(\log 1/\delta)$. □

Considering all possible partitions, how many parameters must we estimate? If, over all partitions there are N distinct blocks, and for each the probability π is learned exactly with probability $1 - \delta/N$, then the probability that any one parameter is learned incorrectly is less than δ. By Eq. 5, with $\epsilon' \equiv \beta/3$, our sample need be no larger than

$$t = \left\lceil \frac{\log(2N/\delta)}{2\epsilon'^2} \right\rceil .$$

(Since we can estimate exactly, the confidence ϵ does not affect the sample size.)

We can bound N as a function of the number of attributes n. Since each box in a model is defined by at most $k-1$ constraints, and each constraint can be chosen in at most $2nr$ ways, the number of distinct boxes N is at most $(2nr)^k$. Hence:

Lemma 4.23 *For any $\delta > 0$, a sample of $O(\log(n/\delta))$ vectors suffices to determine exactly the maximum-likelihood parameters (the π_R's) of all k-RDC partitions, with probability $> 1 - \delta$.*

We can now prove the main result.

Theorem 4.24 *The family k-RDC is data-efficient with respect to any distance measure.*

PROOF: The preceding lemmas show that we can determine the actual probabilities π_R for all the blocks of all possible partitions using a data-efficient sample of vectors. We now show how to use these values to determine a model equivalent to the target M_*.

Let R be a block in some partition. From the proof of Lemma 4.22 we know that all parameters π_R are multiples of some constant β depending on k. Thus π_R/β is an integer for each R. For each model M_1 (consisting of a

partition and the maximum-likelihood probabilities assigned to its blocks), define the *pseudo-likelihood* function $\mathcal{L}(M_1)$ as follows:

$$\mathcal{L}(M_1) = \prod_R \left[\frac{\pi_R}{|R|}\right]^{\pi_R/\beta}$$

Note that $\mathcal{L}$ is a polynomial-size rational number. We claim that the pseudo-likelihood function is maximized precisely on those models equivalent to M_*: i.e., $\mathcal{L}(M_*) \geq \mathcal{L}(M_1)$, with strict inequality whenever $M_* \not\equiv M_1$. From this, we conclude the proof by computing $\mathcal{L}$ for each model and choosing one with the largest such value. Failure can occur only with probability δ, should our parameter estimates be wrong. Since it is a property of every distance measure that the distance $\mathbf{d}$ between equivalent models is zero, the result holds for arbitrary distance functions.

Refer back to the proof of Lemma 4.22 for the notation R, S_j, and X_j. Taking logs, we have

$$\begin{aligned}\log \mathcal{L}(M_1) &= \beta^{-1} \sum_R \pi_R \log(\pi_R/|R|) \\ \log \mathcal{L}(M_*) &= \beta^{-1} \sum_S \pi_S \log(\pi_S/|S|)\end{aligned}$$

We take one term from $\log \mathcal{L}(M_1)$ corresponding to a particular box R and expand this as a sum over the subboxes $X_j = R \cap S_j$.

$$\log \mathcal{L}(M_1)_R = \beta^{-1} \sum_{X_j} \pi_{X_j} \log(\pi_R/|R|).$$

The portion of $\log \mathcal{L}(M_*)$ due to these same sets X_j is

$$\log \mathcal{L}(M_*)_R = \beta^{-1} \sum_{X_j} \pi_{X_j} \log(\pi_{X_j}/|X_j|),$$

since $\pi_{X_j}/|X_j| = \pi_{S_j}/|S_j|$ (just a scale change). Subtracting,

$$\begin{aligned}\log \mathcal{L}(M_*)_R - \log \mathcal{L}(M_1)_R &= \beta^{-1} \sum_{X_j} \pi_{X_j} \log\left(\frac{\pi_{X_j}|R|}{\pi_R|X_j|}\right) \\ &\geq \beta^{-1} \sum_{X_j} \pi_{X_j} \left(1 - \frac{\pi_R|X_j|}{\pi_{X_j}|R|}\right) \\ &= \beta^{-1}(\pi_R - (\pi_R/R) \sum_{X_j} |X_j|) \\ &= 0.\end{aligned}$$

And the inequality above is strictly > unless all $\pi_{X_j}/|X_j|$ are equal — i.e., all vectors in R have the same probability.

Summing over all the boxes R, we have $\log \mathcal{L}(M_*) \geq \log \mathcal{L}(M_1)$, with strict inequality unless the two models are equivalent. The claim follows directly by monotonicity of the log function. □.

As a corollary, we observe that k-RDC is efficiently learnable provided that we can generate and test all N partitions in polynomial (in n) time. (cf. [31]). (We can; details omitted). Furthermore, a small sample is sufficient to learn a model with the smallest number of classes, just by choosing the smallest model maximizing $\mathcal{L}$.

Also, there are several ways to relax the conditions on the model class without sacrificing data efficiency. Most notably, the hard bound k can be replaced by a slowly growing function (i.e., $O(\log n)$) of n, thus allowing both the number of classes to grow (slowly) and the minimum non-zero probability π to get smaller. But while the k-RDC model family is "efficiently" learnable in theory, no generate-and-test algorithm can be considered practical. More seriously, by requiring that all classes in the partition be rectangular – even those containing vectors with zero probability – a large number of such classes must be devoted to "boxing in" vectors which never occur. This situation can be avoided by allowing one set in the partition to be arbitrary (non-rectangular), and requiring that it be the only one with probability $\pi = 0$. Virtually the same argument goes through, provided the conditions imply that all rectangular boxes have probabilities π that are integral multiples of some constant β.

When this condition does not hold, an analysis of a very different kind is needed. Our argument above is very simple because the π's can be learned exactly. When these probabilities can only be closely estimated, the likelihood for any hypothesis can likewise be approximated; but determining how closely d is approximated when the component probabilities are known to within a given accuracy requires quite a bit of detailed analysis of the functions involved. None of this requires any new mathematics, but the analysis is tedious and gives little insight. Furthermore, unlike the preceding result, the analysis depends on the distance function d.

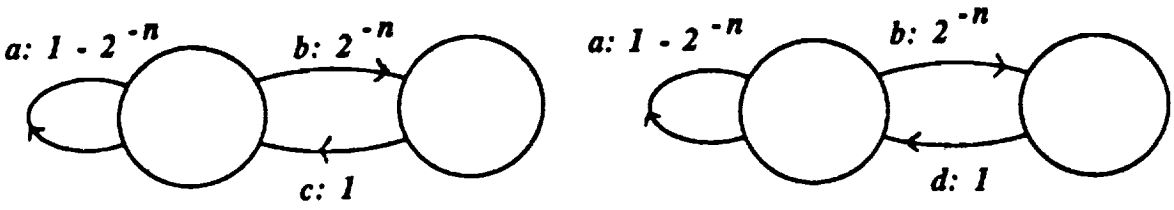


Figure 2: Hard-to-distinguish models.

4.5.3 Markov Models

Our Markov family is described in Example 4.7, with the exception that the set X of input symbols can be any finite set. As in that example, we are interested only in strongly connected chains. We do not assume they are ergodic, since periodic behavior is often important to model.

Finite automata are notoriously difficult to learn in the supervised case [13, 1], so it is not surprising to discover that they are also hard to learn in the unsupervised case. We give a simple proof here for the d_1 distance measure.

Theorem 4.25 The family M_n of Markov models (as defined above) with at most n states and transition probabilities bounded in length by n bits is not data efficient (and hence not efficiently learnable) with respect to d_1.

PROOF: We exhibit two models which a polynomial-size sample cannot distinguish with arbitrarily high probability, even though the two models each have more than ϵ error relative to one another.

Assume an alphabet $A = \{a, b, c, d\}$. Machines M_1 and M_2 are shown in Figure 2. Until a b transition occurs, the predictions for the two models M_1 and M_2 are identical. The probability that no b-transition occurs in t transitions is $(1 - 2^{-n})^t$, and we require t such that this probability is less than δ. Taking logs, we find:

$$t \cdot \log_2(1 - 2^{-n}) < \log_2 \delta.$$

But if t is bounded by a polynomial in n, this condition fails for all sufficiently large n.

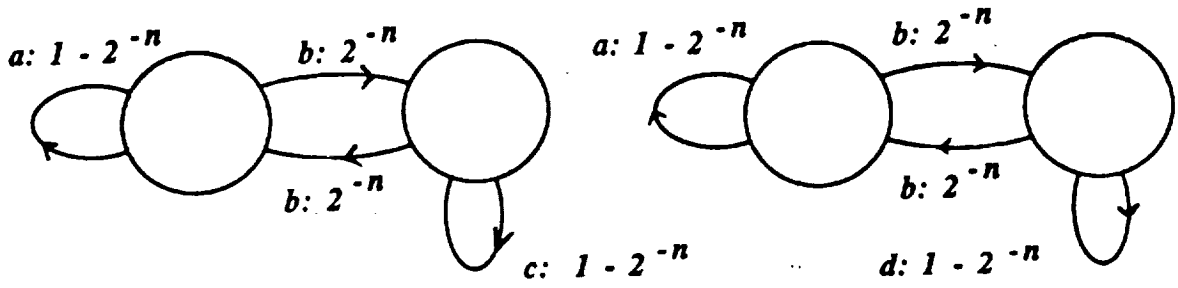


Figure 3: More hard-to-distinguish models.

If no b transition occurs, any algorithm must choose between models M_1 and M_2 on the basis of a string of a's alone. If it chooses M_1 but the actual target is M_2, the error is 1 (the difference in probability for the c transition out of the rightmost state). If it chooses M_2 but the actual target is M_1, the error is again 1 based on the d transition.

The algorithm could still succeed if it proposed some further model M_3 different from either M_1 or M_2, such that $\mathrm{d}(M_3 \mid M_1) < \epsilon$ and $\mathrm{d}(M_3 \mid M_2) < \epsilon$. But we can show that there is no such model. Immediately after the first b transition, suppose M_3 is in a state that predicts a c with probability $p_c < 1$ and a d with probability $p_d < 1$. Necessarily $p_c + p_d \leq 1$, and so if $p_c \geq \frac{1}{2}$ then $p_d \leq \frac{1}{2}$. Thus either $\mathrm{d}(M_3 \mid M_1)$ or $\mathrm{d}(M_3 \mid M_2)$ is at least $\frac{1}{2}$. Hence if $\epsilon < \frac{1}{2}$, no model is within ϵ of both M_1 and M_2. □

The two models in Figure 2 are *not* sufficient to show that M_n is not efficiently learnable with respect to the d_2 measure, because the "distance" between M_1 and M_2 results from the two states on the right which are occupied only about once every 2^{-n} moves. Instead we use the two models in Figure 3, where in each model the two states are each occupied half the time on average. Otherwise, the proof follows a similar line of reasoning as above.

The preceding theorem is based solely on the property that transition probabilities can be exponentially small, thereby making it hard to explore important states of the machine in polynomial time. Unfortunately, merely bounding the transition probabilities to be at least $1/\mathrm{poly}(n)$ is not sufficient to make Markov models efficiently learnable. In Figure 4 we exhibit two models in which all transition probabilities are either 0, $\frac{1}{2}$, or 1.

Even so, exponential expected time is required to reach a state in which the two models make different predictions[8]. Only a bound of $O(\log n)$ on the number of states will remove this problem. And even this may not be enough: the large-numbers limit theorems for Markov chains also depend on the structure of the underlying digraph of the automaton. In order for estimates of the parameters to be reliable, we must be able to reach every state of the process from every other state within a feasibly small expected time. And for general digraphs, this is evidently not known to be possible. All this suggests that, without significant restrictions on the size and nature of the models, a Markov model family will not be efficiently learnable.

5 Conclusions

We have cast the problem of unsupervised learning into a statistical format that differs from that of conventional pattern recognition in its emphasis on efficient prediction and its concern for rapid convergence ("learning"). Starting from definitions of models and families of models classes, we considered four successively stronger concepts of learnability:

1. *learnability in the limit*: disregarding computational cost, a correct model can be identified in finite time.

2. *uniform learnability in the limit*: the same, with a Cauchy convergence criterion that with more data our hypothesized best model gets better according to some fixed distance measure.

3. *data-efficient learnability*: uniformly learnable from a feasibly small amount of data, without consideration of the computational resources required to extract the information from that data.

4. *efficient learnability*: uniformly learnable with a feasibly small data sample that can be processed using feasibly small computing resources.

[8]Thanks to Don Kimber and Nick Littlestone for this example.

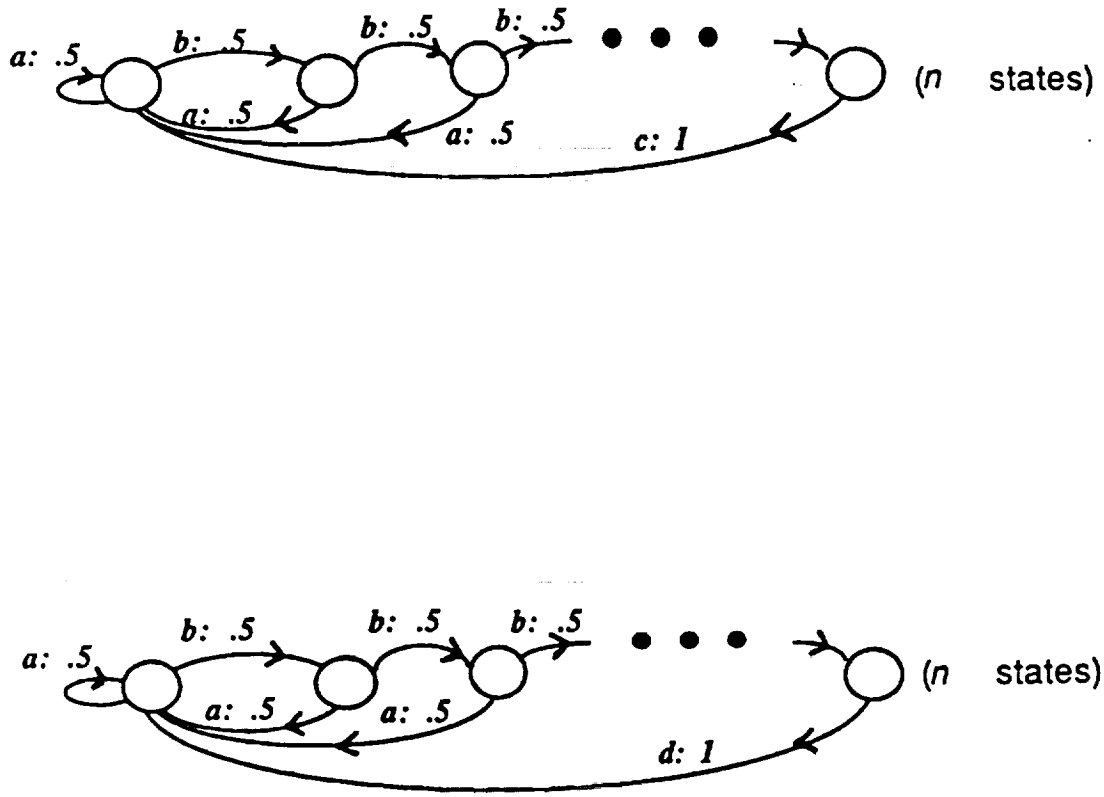


Figure 4: Still more hard-to-distinguish models.

To illustrate these concepts, we have developed and analyzed three model families: Bernoulli families, k-RDC classification families, and a particular Markov family. Much work remains to be done constructing meaningful, expressive, and useful model families, algorithms to learn them, and techniques for analyzing their complexity. Practical learning algorithms need to be "on-line" [16] in the sense that they successively refine their hypotheses as more data arrives, instead of collecting a large clump of data and emitting a single hypothetical model. If the storage resources of the model are bounded, then the predictions of the model may be subject to random-walk behavior as runs of typical and atypical data are received. Related problems concern the performance of such an algorithm when the source of the data is not a model in the target class, and when the best model may change ("drift") with time [27]. And current interest in distributed learning procedures (e.g., [26]) point to the value of considering parallel learning.

While this work was inspired by, and is closely related to, the learnability work based on the Valiant model, it is worth a careful look at the differences in the two models. To that end, let us define a *supervised-modeling problem* as follows. Suppose $\mathcal{F}$ is a set of *classification functions* defined on a finite vector space X. The purpose of these functions is to associate each vector $\mathbf{x}$ with one of a finite set of pre-determined classes $C_1, \ldots, C_k$. However, these functions, rather than assigning a unique class to a vector, output a *probability distribution*: $f(\mathbf{x})$ is a set of k rational values $p_1(\mathbf{x}), \ldots, p_k(\mathbf{x})$ summing to one, with the intended interpretation that $\mathbf{x}$ belongs to class C_i with probability $p_i(\mathbf{x})$. The learning problem is to identify an arbitrary function $f_* \in \mathcal{F}$, given a stream of input examples that have been classified by a teacher. The teacher provides examples of $f_* \in \mathcal{F}$ by choosing a vector $\mathbf{x}$ (not necessarily independently) from X according to some unknown sampling scheme, assigning a classification C_i to x according to the probability distribution $f_*(\mathbf{x})$, and presenting the pair $(\mathbf{x}, C_i)$ to the learning algorithm.

In the special case where the vector space is $\{0,1\}^n$, and there are exactly two classes, and all the probabilities are either zero or one, and the sampling scheme consists of independent trials, we then have the problem of learning Boolean classifiers. The Valiant model is obtained by introducing a distance measure $\mathrm{d}(f_1 \mid f_2)$ and requiring that the distance between the

target and the hypothesized functions be less than ϵ with high probability $(1-\delta)$. The distance (or "error") between f and f_* is the measure of the set $f \triangle f_*$ of vectors on which f and f_* output different classifications, using the probability distribution over X. Let E denote the mathematical expectation of a function with respect to the distribution $\mathbf{P}$ on X. Then since values of f and f_* are 0 or 1,

$$\begin{aligned} \mathrm{d}(f \mid f_*) &= \mathsf{E}[f(\mathbf{x}) - f_*(\mathbf{x})]^2 \\ &= \mathrm{d}_3(f \mid f_*), \end{aligned}$$

the mean-squared error defined above (Example 4.17). So the Valiant model is a highly specialized case of model learning over a vector space $X \times \{0,1\}$, in which the models need not account for the distribution of the X part of the input data, but only for their classification, with the mean-squared error d_3 as the measure of distance between hypotheses. From this perspective, many interesting generalizations of the model – including that of non-independent examples – become possible.

6 Acknowledgments

Discussions with Dana Angluin, Peter Cheeseman, Evan Gamble, David Haussler, Don Kimber, Nick Littlestone, Alex Milosavljevic, Matthew Self, John Stutz, and Manfred Warmuth have been very helpful in developing these ideas. Hamid Berenji, Peter Friedland, Bill Gevarter, Phil Nachtsheim, and my colleagues at NASA Ames Research Center have provided comments that greatly improved the presentation.

References

[1] D. Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 39:337–350, 1978.

[2] Dana Angluin. *Identifying languages from stochastic examples.* Technical Report YALEU/DCS/RR-614, Yale University Dept. of Computer Science, 1988.

[3] M. Aoki. On some convergence questions in bayesian optimization problems. *IEEE Trans. Auto. Contr.*, AC-10:180–182, 1965.

[4] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Stat.*, 37:1554–1563, 1966.

[5] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Ann. Math. Stat.*, 41:164–171, 1970.

[6] L. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.

[7] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. In *Proc. 18th Symposium on Theory of Computing*, pages 273–282, ACM, 1986.

[8] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

[9] P. Cheeseman, J. Kelly, M. Self, J. Stutz, and W. Taylor. Autoclass: a bayesian classification system. In *Proc. Fifth International Conference on Machine Learning*, Kluwer, 1988.

[10] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* J. Wiley, 1973.

[11] B. S. Everitt and D. J. Hand. *Finite Mixture Distributions.* Chapman and Hall, 1981.

[12] W. Feller. *An Introduction to Probability Theory and its Applications, 3rd Edition.* J. Wiley, New York, 1968.

[13] E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.

[14] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[15] D. Haussler, editor. *Machine Learning* (Special Issue on Learning Theory), Kluwer Academic, 1987. (to appear).

[16] D. Haussler, N. Littlestone, and M. Warmuth. Expected mistake bounds for on-line learning algorithms (extended abstract). 1987. (unpublished manuscript).

[17] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Stat. Assoc.*, 58:13–30, 1963.

[18] B.-H. Juang and L. Rabiner. A probabilistic distance measure for hidden markov models. *AT&T Technical Journal*, 64(2):391–407, 1985.

[19] M. Kearns and M. Li et al. On the learnability of boolean formulae. In *Proc. 19th ACM STOC*, 1987.

[20] J. Kiefer and J. Wolfowitz. On the deviations of the empiric distribution function of vector chance variables. *Tr. Am. Math. Soc.*, 85:173–186, 1958.

[21] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Stat.*, 22:79–86, 1951.

[22] J. Levinson, L. Rabiner, and M. Sondhi. An introduction to the application of the theory of probabilistic functions of markov processes in automatic speech recognition. *Bell Sys. Tech. J.*, 62:1035–1074, 1983.

[23] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[24] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:445–471, 1978.

[25] S. Rudich. Inferring the structure of a markov chain from its output. In *Proc. 26th FOCS*, pages 321–325, 1985.

[26] D. E. Rumelhart and J. L. McClelland (eds.). *Parallel Distributed Processing: Explorations in the Microstructures of Cognition (2 Vols.).* MIT Press, 1986.

[27] J. Schlimmer and R. Granger. Beyond incremental processing: tracking concept drift. *Machine Learning*, 1:317–354, 1986.

[28] E. Shapiro. *Algorithmic program debugging.* PhD thesis, Yale University Computer Science Dept., 1982. Published by MIT Press, 1983.

[29] J. E. Shore and R. W. Johnson. Properties of cross-entropy minimization. *IEEE Trans. Inf. Th.*, IT-27:472–482, 1981.

[30] H. Simon. Why should machines learn? In R. S. Michalski et al., editor, *Machine Learning: An AI Approach, Vol. I*, Morgan Kaufmann, 1983.

[31] L. G. Valiant. Learning disjunctions of conjunctions. In *Proceedings of IJCAI*, pages 560–566, IJCAI, 1985.

[32] L. G. Valiant. A theory of the learnable. *C. ACM*, 27:1134–1142, 1984.

[33] S. J. Yakowitz. Unsupervised learning and the identification of finite mixtures. *IEEE Trans. Inf. Th.*, IT-16:330–338, 1970.

[34] S. J. Yakowitz and J. Spragins. On the identifiability of finite mixtures. *Ann. Math. Stat*, 39:209–214, 1968.

[35] T. Y. Young and T. W. Calvert. *Classification, Estimation, and Pattern Recognition.* American Elsevier, 1974.

Appendix

A Proofs of Propositions of Section 4.3

These two propositions follow from more general results available in the statistical literature, but a simple, direct proof is useful when developing and analyzing learning algorithms. For convenience, let us restate the propositions before proving them.

PROPOSITION 4.11: *Let X be a finite set and $\mathcal{M}$ a finite class of models predicting independent inputs over X. Then $\mathcal{M}$ is learnable.*

PROOF: Suppose M' is any such model, and M_* is the model defining the population of X. For any $x \in X$ let p_x be the probability of x according to M_*, and p'_x that according to M'. Suppose we have a large sample of t selections from the population. The particular point x occurs t_x times in the sample; and by the law of large numbers, we know that $t_x/t \to p_x$ almost surely as $t \to \infty$.

The likelihood of the sample according to the model M_* is given by

$$L(X^t \mid M_*) = \prod_{x \in X} {p_x}^{t_x}.$$

Taking logs, negating, and averaging over t, we obtain the *sample entropy* for M_*:

$$H(X^t \mid M_*) = \sum_{x \in X} \frac{t_x}{t} \log(1/p_x).$$

Similarly, the sample entropy for M' is given by

$$H(X^t \mid M') = \sum_{x \in X} \frac{t_x}{t} \log(1/p'_x).$$

We argue that $H(X^t \mid M') \geq H(X^t \mid M_*)$ for almost all t, with equality iff M' and M_* are equivalent. It then follows that $L(X^t \mid M') < L(X^t \mid M_*)$ for almost all t whenever M' is not a correct model, and this enables us to identify the correct model all but finitely often.

To prove the claim, note that

$$\begin{aligned} H(X^t \mid M') - H(X^t \mid M_*) &= \sum_x (t_x/t) \log(p_x/p'_x) \\ &\to \sum_x p_x \log(p_x/p'_x) \text{ as } t \to \infty \end{aligned}$$

in the sense that t_x/t remains within ϵ of p_x for any $\epsilon > 0$. But $\log(p_x/p'_x) \geq 1 - (p'_x/p_x)$, with equality iff $p_x = p'_x$. Thus

$$\begin{aligned} H(X^t \mid M') - H(X^t \mid M_*) &\geq \sum_x p_x(1 - (p'_x/p_x) \\ &= 1 - 1 \\ &= 0. \end{aligned}$$

□

PROPOSITION 4.12 *Let X be a finite set and $\mathcal{M}$ a finite class of Markov chain models. Then $\mathcal{M}$ is learnable in the limit.*

PROOF: Our notation is as follows. A Markov model over the alphabet X has four components:

- a finite set S of states $\{s_1, \ldots, s_n\}$.
- a designated state $s_0 \in S$, called the start state.
- a function $\delta : S \times X \to S$ indicating the next state $\delta(s, a)$ when the character a is received in the current state s.
- for each state s and each character $a \in X$, a probability $p_s(a)$ such that $\sum_{a \in X} p_s(a) = 1$.

Let X^t be a string of length t over the alphabet X. The likelihood $L(X^t \mid M)$ of X^t for a model M is given by the probability $P(X^t \mid M)$. If M is a skeleton (i.e., a chain to which the transition probabilities have not yet been assigned), we obtain the maximum-likelihood parameters as follows. Starting in its initial state, run M through the unique sequence of t transitions determined by X^t, and for each state s, count the total number $t(s)$ of transitions out of that state, as well as the number $t_a(s)$ of transitions from that state on behalf of each character a in the alphabet. States out of which no transitions have occurred may be assigned arbitrary transition probabilities. The ratio $t_a(s)/t(s)$ is a consistent estimator for the probability $p_s(a)$. The likelihood of X^t is given by

$$L(X^t \mid M) = \prod_{\substack{s \in S \\ a \in A}} \left[\frac{t_a(s)}{t(s)}\right]^{t_a(s)}.$$

The sample entropy of M for X^t is thus

$$\hat{H}(X^t \mid M) = -\sum_{s,a} \frac{t_a(s)}{t} \log \frac{t_a(s)}{t(s)}.$$

We assume that the environment is generated by some Markov chain in the family, and that X^t is a string of length t generated by that environment. Let M_* be a Markov chain equivalent to that generating the environment (ambiguously, M_* represents the skeleton for such a chain). S denotes the set of states of M_*. For $s \in S$ and $a \in X$, the notation $t_a(s)$ and $t(s)$ is as defined in the above discussion, pertaining to the skeleton M_*. $p_s(a \mid M_*)$ denotes the actual transition probability out of s on behalf of the character a for the model M_*. Let M_1 be the skeleton of another chain in the family, with R as its state set. For $r \in R$, $t_a(r)$ and $t(r)$ are similarly defined. For purposes of this proof, we should imagine the skeletons M_* and M_1 both being driven in parallel by the string X^t.

The idea is to consider the sample entropy of M_1, in this case for a particular state r, and decompose the entropy of M_* according to the state of M_1.

Fix a state $r \in R$. $\hat{H}_r(X^t \mid M_1)$ denotes that part of the sample entropy contributed while M_1 is making transitions out of state r. (The total sample entropy is the sum over all r in R.) When M_1 is in state r, M_* may be in several states in S. The sample entropy for M_* is likewise the sum of the parts $\hat{H}_r(X^t \mid M_*)$ contributed while M_1 is in r. We shall prove a stronger result than stated in the theorem: that $\hat{H}_r(X^t \mid M_*) \leq \hat{H}_r(X^t \mid M_1)$ for each state $r \in R$, with equality iff all states s occupied by M_* while M_1 is in state r have the same set of transition probabilities. By summing over r and converting from entropy back to likelihood, the theorem obtains directly.

First, we compute the respective sample entropies for M_1 and M_*:

$$\hat{H}_r(X^t \mid M_1) = \frac{1}{t} \sum_a t_a(r) \log \frac{t(r)}{t_a(r)} \tag{6}$$

$$\hat{H}_r(X^t \mid M_*) = \frac{1}{t} \sum_{s \in S} \sum_a t_a(r,s) \log \frac{t(s)}{t_a(s)}, \tag{7}$$

where $t_a(r,s)$ counts the number of a-transitions that occur when both M_1 is in state r and M_* is in state s.

The rest of the proof is just manipulation of these expressions, and passing to the limit. In (7) we write

$$\begin{aligned}\frac{t_a(r,s)}{t} &= \frac{t(r)}{t}\cdot\frac{t(r,s)}{t(r)}\cdot\frac{t_a(r,s)}{t(r,s)}\\ &\equiv C_r\cdot D_{r,s}\cdot\frac{t_a(r,s)}{t(r,s)},\end{aligned}$$

introducing C_r and $D_{r,s}$ for notational convenience. Thus

$$\hat{H}_r(X^t\,|\,M_\bullet) = C_r\sum_s D_{r,s}\sum_a \frac{t_a(r,s)}{t(r,s)}\log\frac{t(s)}{t_a(s)}. \tag{8}$$

As for (6), we can similarly write

$$\begin{aligned}\frac{t_a(r)}{t} &= \frac{t(r)}{t}\cdot\frac{t_a(r)}{t(r)}\\ &= C_r\cdot\sum_s\frac{t_a(r,s)}{t(r)}\\ &= C_r\cdot\sum_s\frac{t(r,s)}{t(r)}\cdot\frac{t_a(r,s)}{t(r,s)}\\ &\equiv C_r\cdot\sum_s D_{r,s}\frac{t_a(r,s)}{t(r,s)}.\end{aligned}$$

Thus

$$\hat{H}_r(X^t\,|\,M_1) = C_r\sum_s D_{r,s}\sum_a\frac{t_a(r,s)}{t(r,s)}\log\frac{t(r)}{t_a(r)}. \tag{9}$$

Note that in the limit, both $t_a(s)/t(s)$ and $t_a(r,s)/t(r,s)$ converge to $p_s(a|M_\bullet)$ (almost-surely). By contrast $t_a(r)/t(r)$, if it approaches any limit, approaches a value that is a linear combination of the probabilities $p_a(s|M_\bullet)$ over the states s occupied at various times by $M_\bullet$ while M_1 is in state r. Thus, in the limit, from (8) and (9) we have

$$\begin{aligned}\hat{H}_r(X^t\,|\,M_1) - \hat{H}_r(X^t\,|\,M_\bullet) &= C_r\sum_s D_{r,s}\sum_a p_s(a|M_\bullet)\log\frac{p_s(X^t\,|\,M_\bullet)t(r)}{t_a(r)}\\ &\geq C_r\sum_s D_{r,s}\sum_a p_s(a|M_\bullet)\left[1-\frac{t_a(r)}{p_s(X^t\,|\,M_\bullet)t(r)}\right] \qquad (10)\\ &= 0.\end{aligned}$$

And in (10) equality holds only if states r and s are equivalent. □

B Bernoulli models and Distance d_2

We show that Bernoulli models are data efficient and efficiently learnable when the d_2 distance measure is used. However, because of the nature of the d_2 function, a different learning algorithm is needed. Suppose λ_* is the actual Bernoulli probability and λ' is the hypothesized model. Then in the limit of large t, the log-likelihood values converge as follows:

$$t^{-1} \log L(X^t \mid \lambda_*) \to \lambda_* \log \lambda_* + (1 - \lambda_*) \log(1 - \lambda_*)$$

$$t^{-1} \log L(X^t \mid \lambda') \to \lambda_* \log \lambda' + (1 - \lambda_*) \log(1 - \lambda').$$

If $0 < \lambda_* < 1$ and $\lambda_1 = 0$ or 1, then

$$\begin{aligned} d_2(\lambda' \mid \lambda_*) &= \lambda_* \log(\lambda_*/\lambda') + (1 - \lambda_*) \log[(1 - \lambda_*)/(1 - \lambda')] \\ &= \infty. \end{aligned}$$

Thus, if λ_* is small and our sample size is also small, we may not observe any 1's; but if we guess $\lambda' = 0$, our error will be infinitely large.

To avoid this, we modify our procedure as follows. Let $a = \epsilon/(1 + 2\epsilon)$, and define ϵ' to be $2^{-n}/3$ if $\epsilon < 2^{-n}$, or $\lfloor (a/2) \cdot 2^n \rfloor \cdot 2^{-n}$ otherwise. As before, we observe at least

$$t = \left\lceil \frac{\log(2/\delta)}{2\epsilon'^2} \right\rceil \tag{11}$$

inputs. Then, if t_1 is the number of 1's in the sample, we shall guess λ' to be the nearest multiple of 2^{-n} to

- (t_1/t), if $\epsilon < 2^{-n}$;
- $\min[(t_1/t) + a/2, 1/2]$ if $2t_1 \leq t$;
- $\max[(t_1/t) - a/2, 1/2]$ if $2t_1 > t$

The first case is identical to that given earlier for d_1, so we need not consider it further. In the other two cases, we are adjusting our estimate λ' toward $\frac{1}{2}$, and away from 0 or 1.

To see why this achieves the desired approximation, assume that $0 \leq \lambda_* \leq \frac{1}{2}$. (The case $\frac{1}{2} \leq \lambda_* \leq 1$ is symmetric to this one and handled

similarly.) The value t has been chosen so that with probability $> 1 - \delta$, $|(t_1/t) - \lambda_*| < a/2$. By adding $a/2$ to the estimate t_1/t, the procedure adjusts the estimate so that $\lambda_* \leq \lambda' < \lambda_* + a$ (with high probability). Writing $\lambda' \equiv \lambda_* + x$, with $0 \leq x < a$, we have

$$\begin{aligned}
\mathbf{d}_2(\lambda' \mid \lambda_*) &= \lambda_* \log(\lambda_*/(\lambda * + x) + (1 - \lambda_*) \log[(1 - \lambda_*)/(1 - \lambda_* - x)] \\
&\leq (1 - \lambda_*) \log[(1 - \lambda_*)/(1 - \lambda_* - x)] \\
&\leq (1 - \lambda_*) \left[\frac{1 - \lambda_*}{1 - \lambda_* - x} - 1 \right] \\
&< \frac{a}{1 - 2a} \\
&= \epsilon.
\end{aligned}$$

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Dates attached	

4. TITLE AND SUBTITLE

Titles/Authors - Attached

5. FUNDING NUMBERS

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Code FIA - Artificial Intelligence Research Branch
Information Sciences Division

8. PERFORMING ORGANIZATION REPORT NUMBER

Attached

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Nasa/Ames Research Center

Moffett Field, CA. 94035-1000

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Available for Public Distribution

Peter Friedland 5/14/92 BRANCH CHIEF

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Abstracts ATTACHED

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT